Program extraction from proofs: induction and coinduction

Ulrich Berger - Swansea

Kanazawa, 8 February 2011

Overview

- Mathematical and formal framework
- Natural and real numbers
- Uniformly continuous functions
- Monadic parsing
- Related work
- Conclusion

Classical mathematics with constructive topping

Axioms:

Any suitable axiom system of classical mathematics (for example ZFC) in a negative formulation, i.e expressing $A \vee B$ as $\neg(\neg A \wedge \neg B)$.

Constructive topping:

Inductive and coinductive definitions as least and greatest fixed points of strictly positive predicate operators (possibly using disjunction).

Intuitionistic first-order logic.

Program extraction

Realisability with uniform interpretation of quantifiers and predicates.

Realizers are elements of a PCA and are denoted by expressions of a lazy functional programming language.

$$\begin{array}{rcll}
a\,\mathbf{r}\,P(\vec{x}) & \equiv & a = \mathrm{Nil} \land P(\vec{x}) & \quad P \text{ primitive} \\
a\,\mathbf{r}\,\forall x\,A(x) & \equiv & \forall x\,a\,\mathbf{r}\,A(x) & \\
a\,\mathbf{r}\,\exists x\,A(x) & \equiv & \exists x\,a\,\mathbf{r}\,A(x) & \\
f\,\mathbf{r}\,(A \to B) & \equiv & \forall a\,(a\,\mathbf{r}\,A \to fa\,\mathbf{r}\,B) & \\
 & \dots & &
\end{array}$$

A suitable formalisation yields Haskell-like extracted programs.

Correctness and typability of extracted programs is proven in a joint paper with M. Seisenberger (to appear).

Unary natural numbers

$\mathbb{R}$ = the real numbers regarded as a primitive predicate with operations and axioms of a real closed field.

$\mathbb{N}$ = the natural numbers as an inductively defined subset of $\mathbb{R}$, i.e. the least subset of $\mathbb{R}$ such that

$$\mathbb{N} \quad = \quad \{0\} \cup \{x + 1 \mid x \in \mathbb{N}\}$$

Formally, $\mathbb{N} := \mu X.\{x \in \mathbb{R} \mid x = 0 \lor \exists y\,(y \in \mathbb{N} \land x = y + 1)\}$.

A first lemma with interesting extracted program:

Lemma 1
$\forall x, y\,(x \in \mathbb{N} \land y \in \mathbb{N} \to x + y \in \mathbb{N})$.

Binary natural numbers

Binary digits:

$$\mathbb{D} \quad := \quad \{0,1\} = \{x \in \mathbb{R} \mid x = 0 \lor x = 1\}$$

Binary natural numbers:

$$\mathbb{B} \quad := \quad \mu X.\{x \in \mathbb{R} \mid x \in \mathbb{D} \lor \exists y \in X\, \exists d \in \mathbb{D}\, (y > 0 \land x = 2y + d)\}$$

Lemma 2

$\forall x, y\, (x \in \mathbb{B} \land y \in \mathbb{B} \to x + y \in \mathbb{B})$.

Lemma 3

$\mathbb{B} = \mathbb{N}$ (i.e. $\mathbb{B} \subseteq \mathbb{N}$ and $\mathbb{N} \subseteq \mathbb{B}$).

Approaching real numbers coinductively

$\mathrm{SD} := \{0, 1, -1\}$

$\mathbb{I} := [-1, 1] = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$

$\mathbb{I}_d := [\frac{d-1}{2}, \frac{d+1}{2}]$

We define $\mathrm{C}_0 \subseteq \mathbb{R}$ coinductively by

$$\mathrm{C}_0 \quad := \quad \nu X.\{x \in \mathbb{R} \mid \exists d \in \mathrm{SD}\,(x \in \mathbb{I}_d \wedge \exists y \in X\, x = \frac{y+d}{2})\}$$

Idea: locate x in a subinterval $\mathbb{I}_d$ (and output d), zoom in, repeat.

A realiser of $x \in \mathrm{C}_0$ is an infinite stream of signed digits $a = a_0 : a_1 : \ldots$ representing x, i.e. $x = \sum_i a_i 2^{-(i+1)}$.

C_0 vs. Cauchy sequences

Lemma 4

$x \in \mathrm{C}_0 \quad \text{iff} \quad \forall n \in \mathbb{N}\, \exists q \in \mathbb{Q} \cap \mathbb{I}\, |x - q| \leq 2^{-n}$.

From a proof of Lemma 4 one extracts programs translating between the signed-digit- and the Cauchy-representation.

Extracting exact real arithmetic

Theorem 2 If $x, y \in \mathrm{C}_0$ then $\frac{x+y}{2} \in \mathrm{C}_0$.

Theorem 3 If $x, y \in \mathrm{C}_0$ then $xy \in \mathrm{C}_0$.

From these theorems one extracts implementations of addition and multiplication w.r.t. the signed digit representation.

Similar implementations were studied by Edalat, Potts, Heckmann, Escardo, Ciaffaglione, Gianantonio, e.t.c.

The difference is that we *extract* the programs
– together with their correctness proofs.

Approaching real functions coinductively

Let f, g, h range over functions from $\mathbb{I}$ to $\mathbb{I}$ and F, G over sets of such functions.

Set $\mathrm{av}_d(x) := \frac{d+x}{2}$.

We define $\mathrm{C}_1 \subseteq \mathbb{I}^{\mathbb{I}}$ by a nested inductive/coinductive definition:

$$\begin{aligned} \mathrm{C}_1 = \nu F.\mu G.\{h \mid\ & \exists e \in \mathrm{SD}\, \exists f \in F\, (h = \mathrm{av}_e \circ f) \vee \\ & \forall d \in \mathrm{SD}\, h \circ \mathrm{av}_d \in G\} \end{aligned}$$

Idea:

- If possible, locate $h[\mathbb{I}]$ in a subinterval $\mathbb{I}_e$ (and output e), zoom in, repeat.
- Otherwise, try all possible input digits d and 'restrict' h to $\mathbb{I}_d$, repeat until eventually the first case applies.

Memo tries for continuous functions

Theorem 4 *h* is uniformly continuous iff $h \in \mathrm{C}_1$.

From the proof of this theorem one extracts programs translating between realisers of "*f* is continuous" (where continuity has to be defined in a constructively meaningful way) and realisers of "$f \in \mathrm{C}_1$".

What is a realiser of "$f \in \mathrm{C}_1$"?

It is a finitely branching non-wellfounded tree describing when *f* emits and absorbs digits. I.p. it is a *data structure*, not a function.

Similar trees have been studied by P. Hancock, D. Pattinson, N. Ghani.

Extracting memoized exact real arithmetic

The definition of $\mathrm{C}_1 \subseteq \mathbb{I}^{\mathbb{I}}$ can be generalised to $\mathrm{C}_n \subseteq \mathbb{I}^{(\mathbb{I}^n)}$.

Theorem 5 The average function lies in C_2.

Theorem 6 Multiplication lies in C_2.

From Theorems 5,6 one extracts implementations of addition and multiplication as memo-tries (relation to work by Hinze and Altenkirch?)

Experiments show considerable speed-up when sampling "hard" functions (e.g. high iterations of the logistic map) on a very fine grid.

Problems in Coq with the nested inductive/coinductive definition of C_1:

Coq accepts the definition, but rejects the attempted coinductive proofs as not formally guarded.

Integration

Let $\int f$ denote the definite integral $\int_{-1}^{1} f(x)\mathrm{d}x$.

We assume the following "axioms" about $\int f$:

(a) $\int f = \frac{1}{2}\int(\mathrm{va}_d \circ f) + d \quad$ where $\mathrm{va}_d(x) := 2x - d$.

(b) $\int f = \frac{1}{2}(\int(f \circ \mathrm{av}_{-1}) + \int(f \circ \mathrm{av}_1))$.

Theorem 7 If $f \in \mathrm{C}_1$, then $\forall n \in \mathbb{N}\, \exists q \in \mathbb{Q}\, |\int f - q| \leq 2^{-n}$.

The extracted program has some similarity with A. Simpson's, but is more efficient because the functions to be integrated are represented differently.

What have we achieved?

- Programs with correctness proofs extracted (some new, some more efficient).
- Simple formalisation: Abstract classical reals, no streams, no trees, ...
- Simple proofs, e.g. in Coq.
- In some cases first hacking the program and then verifying it would have been much harder than the extraction of the program from a proof (for example, the proof that digit spaces have finite products).

Finite sets

Let $\mathcal{P}(X)$ be the classical powerset of X.

Define $\mathcal{P}_{<\omega}(X) \subseteq \mathcal{P}(X)$ by a constructive inductive definition:

(i) $\emptyset \in \mathcal{P}_{<\omega}(X)$
(ii) If $E \in \mathcal{P}_{<\omega}(X)$ and $x \in X$, then $\{x\} \tilde{\cup} E \in \mathcal{P}_{<\omega}(X)$

where $A \tilde{\cup} B := \{x \mid x \in A \tilde{\vee} x \in B\}$ and $\tilde{\vee}$ is classical disjunction (we assume comprehension for classical properties, hence $A \tilde{\cup} B$ exists).

Finite sets: formal definition

Let $\mathcal{X}$ range over subsets of $\mathcal{P}(X)$.

$$\mathcal{P}_{<\omega}(X) = \mu\mathcal{X}.\{F \in \mathcal{P}(X) \mid F = \emptyset \vee \exists x \in X\, \exists E \in \mathcal{X}\; F = \{x\} \tilde{\cup} E\}$$

A realiser of "$E \in \mathcal{P}_{<\omega}(X)$" is a finite list $[a_1, \ldots, a_n]$ such that a_i realises "$x_i \in X$" and $E = \{x_1, \ldots, x_n\}$.

In particular, if X is a "concrete" set, that is, its elements realise themselves, then a realiser of "$E \in \mathcal{P}_{<\omega}(X)$" is simply a listing of the elements of E.

Labelled transition systems

Let S, A be sets (states and labels). For simplicity let's assume both are concrete.

$\mathrm{LTS}_{S,A} := \mathcal{P}(S \times A \times S)$.

Finitely branching LTS Let $P \in \mathrm{LTS}_{S,A}$.

$$\mathrm{FB}_{S,A}(P) :\equiv \forall s \in S \, P(s) \in \mathcal{P}_{<\omega}(A \times S)$$

where $P(s) := \{(a,t) \mid (s,a,t) \in P\}$.

A realiser of "$\mathrm{FB}_{S,A}(P)$" is a function $p \colon S \to [A \times S]$ such that $p(s)$ is a listing of all (a,t) with $P(s,a,t)$.

Constructing finitely branching LTS

$\mathrm{return}(a) := \{(s, a, s) \mid s \in S\}$ (for $a \in A$).
$\mathrm{fail} := \emptyset$

Lemma

(a) $\mathrm{FB}_{S,A}(\mathrm{return}(a))$
(b) $\mathrm{FB}_{S,A}(\mathrm{fail})$
(c) If $\mathrm{FB}_{S,A}(P)$ and $\mathrm{FB}_{S,A}(Q)$, then $\mathrm{FB}_{S,A}(A \mathbin{\tilde{\cup}} B)$

Sequencing finitely branching LTS

If $P \in \mathrm{LTS}_{S,A}$ and $Q_a \in \mathrm{LTS}_{S,B}$ for $a \in A$, then we define

$$P >>= Q := \{(s, b, t) \mid \exists a, r\,(P(s, a, r) \wedge Q_a(r, b, t))\}$$

Lemma If $\mathrm{FB}_{S,A}(P)$ and $FB_{S,B}(Q_a)$ for all $a \in A$, then $FB_{S,B}(P >>= Q)$.

From these lemmas the corresponding monadic parsers and parser combinators can be extracted.

For more parser combinators the set S must be instantiated by the set of strings.

What have we achieved?

- The well-known parser combinators by Hutton/Meijer have been extracted – with correctness and in particular termination proofs!
- In the (source) proofs no lists or higher-order functions occur.

Related work on realisability for (co)induction

M. Tatsuta, Realizability of Monotone Coinductive Definitions and Its Application to Program Synthesis. Proc. MPC, LNCS 1422, 338–364, 1998

F. Miranda-Perea, Realizability for Monotone Clausular (Co)Inductive Definitions, ENTCS 123, 179–193, 2005

H. Schwichtenberg, Minlog system, `http://www.mathematik.uni-muenchen.de/∼minlog/minlog/`

B., Realisability for induction and coinduction, Proceedings of Computability and Complexity in Analysis (CCA), 2009

Case studies in program extraction

- NbE extracted from Tait's SN proof for the simply typed λ-calculus.
  Berghofer (Isabelle), Letouzey (Coq), Schwichtenberg, B. (Minlog).
- Program extracted from Nash-Williams classical proof of Dickson's Lemma and Higman's Lemma
  Seisenberger, B. (Minlog).
- Programs extracted from Intermediate Value Theorem and Inverse Function Theorem for continuous real functions.
  Schwichtenberg (Minlog).

Proof-theoretic strength

- Without classical axioms our formal system is an intuitionistic first-order version of the μ-calculus (or fixed point logic).
- With classical logic the system has the proof-theoretic strength of Π^1_2-comprehension (Möllerfeld 2007).
- The intuitionistic version, however with non-strictly positive inductive definitions, has the same strength.

  [S. Tupailo, On the intuitionistic strength of monotone inductive definitions,JSL 69(3), 790–798, 2004]

Conclusion

- Program extraction turned out to be very helpful (not a burden) in the example areas covered.
- Can we apply it to areas that are of less mathematical nature? (parsing is a start)
- Can we address resource issues?
- Coq and Minlog work well, generally, but there are problems with nested inductive coinductive definitions.

What I'm currently working on

Consider *pure data types*, A, B, i.e. types built from 1, $+$, $\times$, μ, ν.

- Transform the function type

$$A \to B$$

  into a pure data type.

This problem was solved by Hinze, Altenkirch (and possibly others) for the case that A is *inductive*, i.e. contains no ν.

A partial solution for the general case was given by Ghani, Hancock and Pattinson.

The problem is relevant for modelling efficiently continuous real functions of several variables and possibly higher type functionals over the reals (e.g. integration).